Big-Step Structural Operational Semantics (Big-Step SOS)

- Gilles Kahn (1987), under the name *natural semantics*
- Also known as *relational semantics*, or *evaluation semantics*
- One can regard a big-step SOS as a recursive interpreter, telling for a fragment of code and state what it evaluates to
- *Configuration*: tuple containing code and semantic ingredients
  - E.g., $\langle a_1, \sigma \rangle \quad \langle a_1 + a_2, \sigma \rangle \quad \langle i_1 \rangle \quad \langle i_1 +_{Int} i_2 \rangle \quad \langle \sigma \rangle$
- *Sequent*: Pair of configurations, to be *derived* or *proved*
  - E.g., $\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle$
- *Rule*: Tells how to derive a sequent from others
  - E.g.,

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 + a_2, \sigma \rangle \Downarrow \langle i_1 +_{Int} i_2 \rangle}$$

Big-Step SOS of IMP - Arithmetic

$$\langle i, \sigma \rangle \Downarrow \langle i \rangle \qquad \text{(BigStep-Int)}$$

State lookup

$$\langle x, \sigma \rangle \Downarrow \langle \sigma(x) \rangle \qquad \text{(BigStep-Lookup)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \ + \ a_2, \sigma \rangle \Downarrow \langle i_1 \ +_{Int} \ i_2 \rangle} \qquad \text{(BigStep-Add)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \ / \ a_2, \sigma \rangle \Downarrow \langle i_1 \ /_{Int} \ i_2 \rangle} \text{, where } i_2 \neq 0 \qquad \text{(BigStep-Div)}$$

Big-Step SOS of IMP - Boolean

$$\langle t, \sigma \rangle \Downarrow \langle t \rangle \qquad \text{(BigStep-Bool)}$$

$$\frac{\langle a_1, \sigma \rangle \Downarrow \langle i_1 \rangle \qquad \langle a_2, \sigma \rangle \Downarrow \langle i_2 \rangle}{\langle a_1 \texttt{ <= } a_2, \sigma \rangle \Downarrow \langle i_1 \ \leq_{Int} \ i_2 \rangle} \qquad \text{(BigStep-Leq)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle}{\langle \texttt{not } b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad \text{(BigStep-Not-True)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle \texttt{not } b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle} \qquad \text{(BigStep-Not-False)}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle b_1 \texttt{ and } b_2, \sigma \rangle \Downarrow \langle \texttt{false} \rangle} \qquad \text{(BigStep-And-False)}$$

$$\frac{\langle b_1, \sigma \rangle \Downarrow \langle \texttt{true} \rangle \qquad \langle b_2, \sigma \rangle \Downarrow \langle t \rangle}{\langle b_1 \texttt{ and } b_2, \sigma \rangle \Downarrow \langle t \rangle} \qquad \text{(BigStep-And-True)}$$

Big-Step SOS of IMP - Statements

$$\langle \texttt{skip}, \sigma \rangle \Downarrow \langle \sigma \rangle \qquad \text{(BigStep-Skip)}$$

State update

$$\frac{\langle a, \sigma \rangle \Downarrow \langle i \rangle}{\langle x := a, \sigma \rangle \Downarrow \langle \sigma[i/x] \rangle} \qquad \text{(BigStep-Asgn)}$$

$$\frac{\langle s_1, \sigma \rangle \Downarrow \langle \sigma_1 \rangle \qquad \langle s_2, \sigma_1 \rangle \Downarrow \langle \sigma_2 \rangle}{\langle s_1 ; s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle} \qquad \text{(BigStep-Seq)}$$

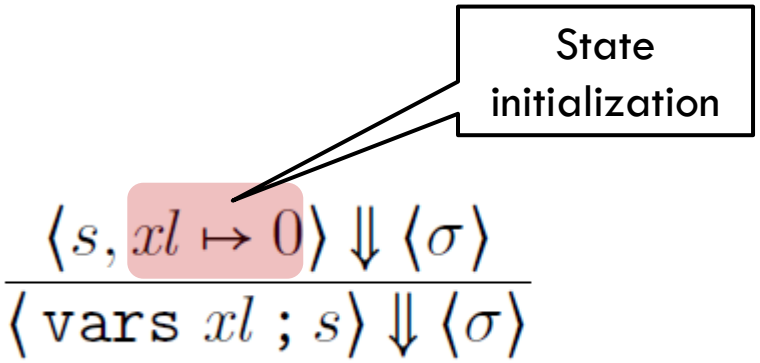
$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle \qquad \langle s_1, \sigma \rangle \Downarrow \langle \sigma_1 \rangle}{\langle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2, \sigma \rangle \Downarrow \langle \sigma_1 \rangle} \qquad \text{(BigStep-If-True)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle \qquad \langle s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle}{\langle \texttt{if } b \texttt{ then } s_1 \texttt{ else } s_2, \sigma \rangle \Downarrow \langle \sigma_2 \rangle} \qquad \text{(BigStep-If-False)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{false} \rangle}{\langle \texttt{while } b \texttt{ do } s, \sigma \rangle \Downarrow \langle \sigma \rangle} \qquad \text{(BigStep-While-False)}$$

$$\frac{\langle b, \sigma \rangle \Downarrow \langle \texttt{true} \rangle \qquad \langle s ; \texttt{while } b \texttt{ do } s, \sigma \rangle \Downarrow \langle \sigma' \rangle}{\langle \texttt{while } b \texttt{ do } s, \sigma \rangle \Downarrow \langle \sigma' \rangle} \qquad \text{(BigStep-While-True)}$$

Big-Step SOS of IMP - Programs



$$\frac{\langle s, xl \mapsto 0 \rangle \Downarrow \langle \sigma \rangle}{\langle \texttt{vars}\ xl\ \texttt{;}\ s \rangle \Downarrow \langle \sigma \rangle}$$

(BIGSTEP-VARS)

Big-Step Rule Instances

- Rules are schemas, allowing recursively enumerable many instances; side conditions filter out instances
  - E.g., these are correct instances of the rule for division

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle 2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 2\rangle}{\langle x/2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 4\rangle}$$

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle 2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 4\rangle}{\langle x/2, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 2\rangle}$$

The second may look suspicious, but it is not. Normally, one should never be able to apply it, because one cannot prove its hypotheses

  - However, the following is *not* a correct instance (no matter what ? is):

$$\frac{\langle x, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 8\rangle \qquad \langle y, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle 0\rangle}{\langle x/y, (x \mapsto 8, y \mapsto 0)\rangle \Downarrow \langle ?\rangle}$$

Big-Step SOS Derivation

The following is a valid proof derivation, or proof tree, using the big-step SOS proof system of IMP above.
Suppose that x and y are identifiers and $\sigma(x)=8$ and $\sigma(y)=0$.

$$\dfrac{\dfrac{}{\langle x,\sigma\rangle \Downarrow \langle 8\rangle} \qquad \dfrac{\dfrac{\dfrac{}{\langle y,\sigma\rangle \Downarrow \langle 0\rangle} \qquad \dfrac{}{\langle x,\sigma\rangle \Downarrow \langle 8\rangle}}{\langle y/x,\sigma\rangle \Downarrow \langle 0\rangle} \qquad \dfrac{}{\langle 2,\sigma\rangle \Downarrow \langle 2\rangle}}{\langle y/x+2,\sigma\rangle \Downarrow \langle 2\rangle}}{\langle x/(y/x+2),\sigma\rangle \Downarrow \langle 4\rangle}$$

Big-Step SOS for Type Systems

- Big-Step SOS routinely used to define type systems for programming languages
- The idea is that a fragment of code c, in a given *type environment* Γ, can be assigned a certain type τ. We typically write

instead of

$$\Gamma \vdash c : \tau$$

$$\langle c, \Gamma \rangle \Downarrow \langle \tau \rangle$$

- Since all variables in IMP have integer type, Γ can be replaced by a list of untyped variables in our case. In general, however, a type environment Γ contains *typed variables*, that is, pairs "$x : \tau$".

Typing Arithmetic Expressions

$$xl \vdash i : int \qquad \text{(BigStepTypeSystem-Int)}$$

$$(xl, x, xl') \vdash x : int \qquad \text{(BigStepTypeSystem-Lookup)}$$

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 \ + \ a_2 : int} \qquad \text{(BigStepTypeSystem-Add)}$$

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 \ / \ a_2 : int} \qquad \text{(BigStepTypeSystem-Div)}$$

Typing Boolean Expressions

$$xl \vdash t : bool \qquad \text{(BigStepTypeSystem-Bool)}$$

$$\frac{xl \vdash a_1 : int \qquad xl \vdash a_2 : int}{xl \vdash a_1 \texttt{ <= } a_2 : bool} \qquad \text{(BigStepTypeSystem-Leq)}$$

$$\frac{xl \vdash b : bool}{xl \vdash \texttt{not } b : bool} \qquad \text{(BigStepTypeSystem-Not)}$$

$$\frac{xl \vdash b_1 : bool \qquad xl \vdash b_2 : bool}{xl \vdash b_1 \texttt{ and } b_2 : bool} \qquad \text{(BigStepTypeSystem-And)}$$

Typing Statements

$$xl \vdash \texttt{skip} : stmt \qquad \text{(BigStepTypeSystem-Skip)}$$

$$\frac{(xl, x, xl') \vdash a : int}{(xl, x, xl') \vdash (x := a) : stmt} \qquad \text{(BigStepTypeSystem-Asgn)}$$

$$\frac{xl \vdash s_1 : stmt \qquad xl \vdash s_2 : stmt}{xl \vdash s_1 ; s_2 : stmt} \qquad \text{(BigStepTypeSystem-Seq)}$$

$$\frac{xl \vdash b : bool \qquad xl \vdash s_1 : stmt \qquad xl \vdash s_2 : stmt}{xl \vdash \texttt{if}\ b\ \texttt{then}\ s_1\ \texttt{else}\ s_2 : stmt} \qquad \text{(BigStepTypeSystem-If)}$$

$$\frac{xl \vdash b : bool \qquad xl \vdash s : stmt}{xl \vdash \texttt{while}\ b\ \texttt{do}\ s : stmt} \qquad \text{(BigStepTypeSystem-While)}$$

Typing Programs

$$\frac{xl \vdash s : stmt}{\vdash \texttt{vars}\ xl\ \texttt{;}\ s : pgm}$$

(BigStepTypeSystem-Vars)

Big-Step SOS Type Derivation

Like the big-step rules for the concrete semantics of IMP, the ones for its type system are also rule schemas. We next show a proof derivation for the well-typeness of an IMP program that adds all the numbers from 1 to 100:

$$\dfrac{\dfrac{tree_1 \qquad \dfrac{tree_2 \qquad tree_3}{\texttt{n,s} \vdash (\texttt{while not(n<=0) do (s:=s+n; n:= n+-1)}) : stmt}}{\texttt{n,s} \vdash (\texttt{n:=100; s:=0; while not(n<=0) do (s:=s+n; n:=n+-1)}) : stmt}}{\vdash (\texttt{vars n,s; n:=100; s:=0; while not(n<=0) do (s:=s+n; n:=n+-1)}) : pgm}$$

where

Big-Step SOS Type Derivation

$$tree_1 = \left\{ \dfrac{\dfrac{\dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash 100 : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{n}:=100) : stmt} \quad \dfrac{\dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash 0 : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{s}:=0) : stmt}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{n}:=100;\ \mathtt{s}:=0) : stmt} \right.$$

$$tree_2 = \left\{ \dfrac{\dfrac{\dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash \mathtt{n} : int} \quad \dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash 0 : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{n}<=0) : bool}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{not}\,(\mathtt{n}<=0)) : bool} \right.$$

Big-Step SOS Type Derivation

$$tree_3 \quad = \quad \left\{ \dfrac{\dfrac{\dfrac{\dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash \mathtt{s} : int} \quad \dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash \mathtt{n} : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{s+n}) : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{s:=s+n}) : stmt} \quad \dfrac{\dfrac{\dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash \mathtt{n} : int} \quad \dfrac{\cdot}{\mathtt{n},\mathtt{s} \vdash \mathtt{-1} : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{n+-1}) : int}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{n:=\ n+-1}) : stmt}}{\mathtt{n},\mathtt{s} \vdash (\mathtt{s:=s+n;\ n:=\ n+-1}) : stmt} \right.$$

Big-Step SOS in Rewriting Logic

- Any big-step SOS can be associated a rewrite logic theory (or, equivalently, a Maude module)
- The idea is to associate to each big-step SOS rule

$$\frac{C_1 \Downarrow R_1 \qquad C_2 \Downarrow R_2 \qquad \ldots \qquad C_n \Downarrow R_n}{C \Downarrow R}$$

a rewrite rule

$$\overline{C} \to \overline{R} \;\; \textbf{if} \;\; \overline{C_1} \to \overline{R_1} \;\wedge\; \overline{C_2} \to \overline{R_2} \;\wedge\; \ldots \;\wedge\; \overline{C_n} \to \overline{R_n}$$

(over-lining means “algebraization”)

Big-Step SOS of IMP in Maude

- See file **`imp-semantics-bigstep.maude`**
- See file **`imp-type-system-bigstep.maude`**